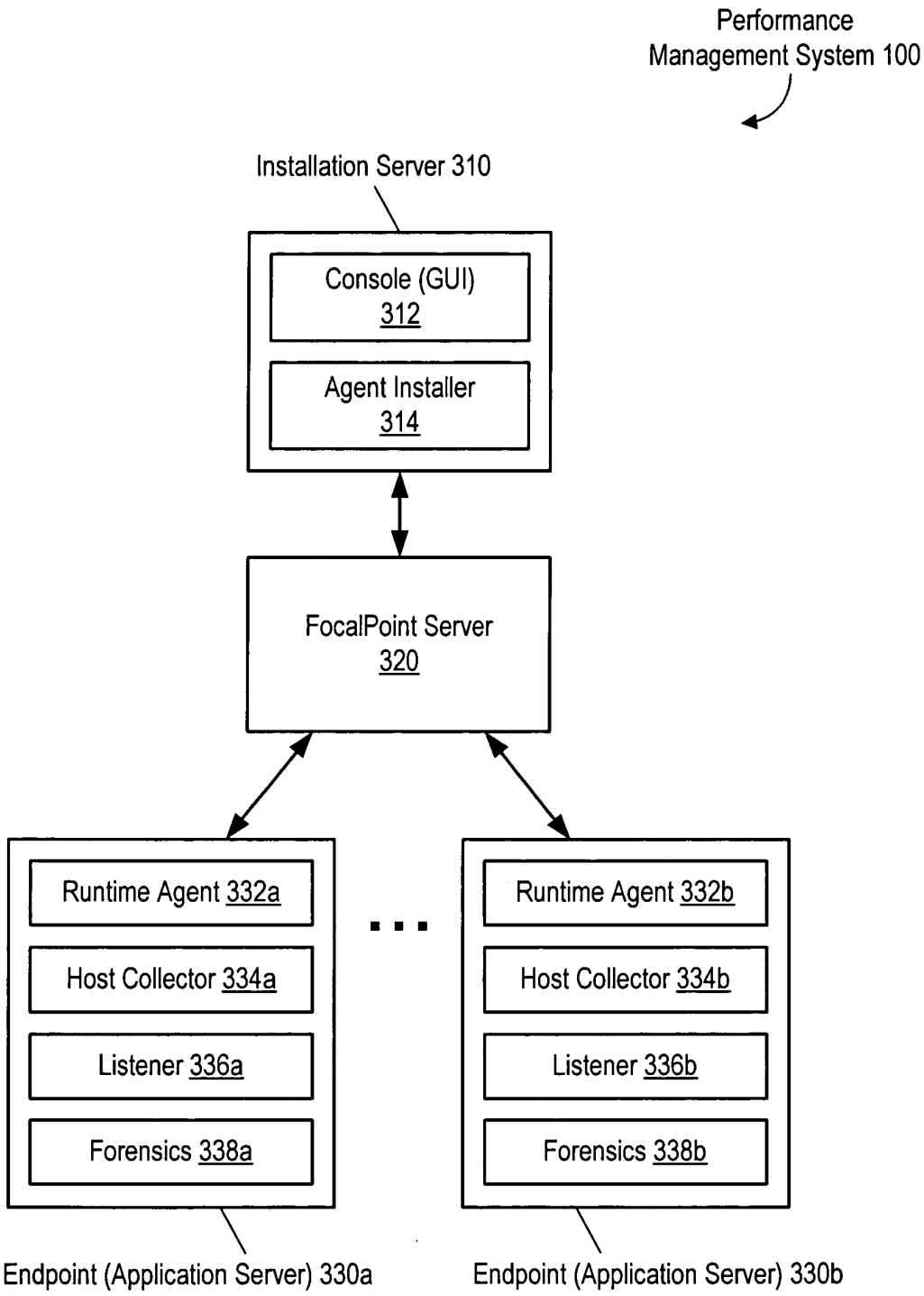
Performance Management System 100
Installation Server 310
Console (GUI) 312
Agent Installer 314
FocalPoint Server 320
Runtime Agent 332a
Host Collector 334a
Listener 336a
Forensics 338a
Runtime Agent 332b
Host Collector 334b
Listener 336b
Forensics 338b
Endpoint (Application Server) 330a
Endpoint (Application Server) 330b

Figure 1

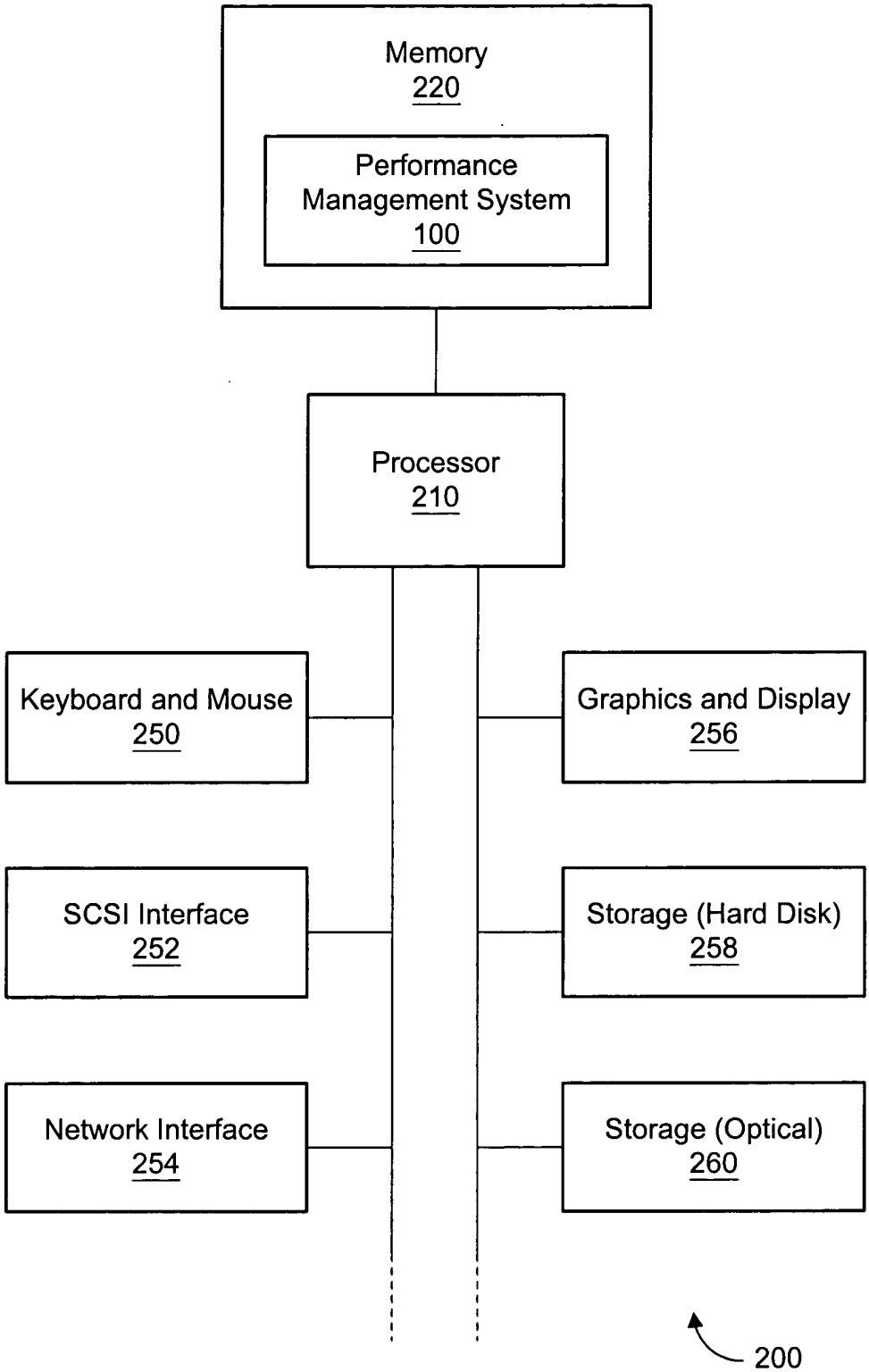
Memory
220
Performance
Management System
100
Processor
210
Keyboard and Mouse
250
Graphics and Display
256
SCSI Interface
252
Storage (Hard Disk)
258
Network Interface
254
Storage (Optical)
260
200

Figure 2

Metric	Description
allocated-memory	The amount of memory in bytes that a program has dynamically allocated.
allocated-objects	The count of memory objects that a program has dynamically allocated.
cpu-utilization	The fraction of the total time spent actually computing or processing data, as opposed to waiting for users, other programs, etc.
current-threads	The number of threads that currently exist within a program.
file-handle-count	The number of opened files within a program.
heap-size	Heap size includes live memory plus any and all overhead associated with the performance management system's management of the heap (data structures, fragmentation, and other internal use of memory).
live-memory	The amount of memory, in bytes, that has been dynamically allocated and has not been freed. It includes both reachable memory and leaked memory that has not been collected.
live-objects	The count of objects that have been dynamically allocated and have not been freed. It includes both reachable objects and leaked objects that have not been collected.
number-of-heaps	The count of how many heaps are in use by a process.
physical-memory	Real or actual memory (e.g., RAM), in bytes, that the operating system has allocated to this application.
reachable-memory	Memory, in bytes, that the program can actually access legitimately through program variables. Thus it does not include leaked memory.
system-time	The amount of time, in seconds, that a program has spent in the "system", as opposed to application code.
threads-created	The total number of threads created in a program. This includes both threads that currently exist and threads that have been destroyed.
total-virtual-memory	The maximum amount of committed virtual memory that can be in use on a computer, in bytes.
user-cpu-utilization	"CPU utilization", given in seconds, is typically separated into multiple components. The "user" portion is the amount of time spent in the program code itself, as a fraction of the total time, as opposed to the operating system.
user-time	The amount of time, in seconds, that a program has spent in application code, as opposed to the operating system.
virtual-memory	Memory, in bytes, that the operating system has allocated to a program. This memory may be "physical" (backed by RAM), or swapped out (backed by disk), or unallocated (not backed by anything), if it has not been referenced.
wall-time	The amount of actual time, in seconds, that a program has been alive.

Figure 3A

Metric	Description
collected-objects	The count of leaked objects that have been automatically freed by the Runtime Agent.
collected-memory	The amount of leaked memory, in bytes, that has been automatically freed by the Runtime agent. Note: This metric is reported in the console as Recovered Memory.
corrected-overwrites	The count of memory overwrites that have been corrected by the Runtime Agent.
corrected-write-after-free	The count of memory write-after-frees that have been corrected by the Runtime Agent.
exception-count	The number of exceptions that the program has raised.
heap-lock-contentions	Heap lock contentions occur when an acquisition attempt is made, but some other thread of the process is in the middle of making a heap request, and so this thread must wait. A large number of heap lock contentions (i.e., a high rate) indicates that dynamic memory allocation is a performance or scalability problem.
leaked-memory	Memory, in bytes, that has been allocated by a program, not explicitly freed, but which is no longer referenced. Leaked memory is thus wasted, since it still contributes to program memory requirements, but cannot be used.
leaked-objects	The count of objects that have been allocated by a program, not explicitly freed, but which are no longer referenced.
multiple-frees	A multiple-free occurs when a program explicitly frees the same object more than once. This can lead to corruption of the memory allocation system, and hence of program memory areas. "Multiple-frees" is the number of such instances.
total-lock-contentions	A program "lock" is a synchronization resource used to mediate access from multiple threads or CPUs. If one thread has locked a resource, and another thread attempts to access it, the second thread is said to "contend" for the lock. A large number of lock contentions (i.e., a high rate) may hurt performance or scalability. "Total-lock-contentions" is the count of contentions from all types of locks.
uncorrected-overwrites	A memory overwrite occurs when a program stores a value beyond the actual bounds of a memory object. Typically these are uncorrected and could corrupt other program memory. "Uncorrected-overwrites" is the number of such instances.
uncorrected-write-after-free	A memory write-after-free occurs when a program explicitly frees a dynamically-allocated object, then continues to store into it. Typically these are uncorrected, and could corrupt other program memory. "Uncorrected-write-after-free" is the number of such instances.
wild-frees	A wild free occurs when a program explicitly frees some memory location that was not dynamically allocated in the first place. This can lead to corruption of the memory allocation system, and hence of program memory areas. "Wild-frees" is the number of such instances.

Figure 3B

Metric	Description
cpu-count	The number of CPUs in a computer.
file-handle-count-limit	The maximum number of open files within a program.
physical-memory-limit	The maximum amount of RAM, in bytes, that the operating system will allocate to a particular process.
physical-memory-size	The total amount of RAM, in bytes, on a particular computer.
used-virtual-memory	The total virtual memory committed for use across all programs on a computer, in bytes.
virtual-memory-limit	The maximum amount of virtual memory, in bytes, that a program could allocate.

Figure 3C

Metric	Description
collections	The count of how many collections have been run by the Runtime Agent.

Figure 3D

Metric	Description
-rate	Change over each interval (per minute). Typically rates are presented across the last FocalPoint polling interval.
-trend	Linear regression over process lifetime (per minute)
-max	Maximum over lifetime
-avg	Average over lifetime

Figure 3E

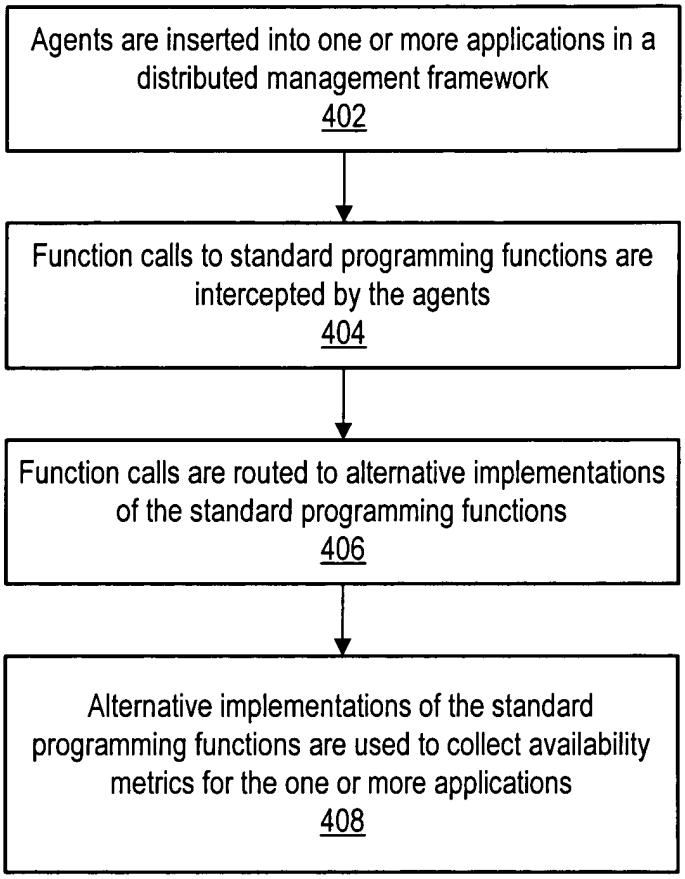
Agents are inserted into one or more applications in a distributed management framework
402
Function calls to standard programming functions are intercepted by the agents
404
Function calls are routed to alternative implementations of the standard programming functions
406
Alternative implementations of the standard programming functions are used to collect availability metrics for the one or more applications
408

Figure 4

Instrumentation options	
-v	Print out version information and exit.
-h	Print out help message and exit.
-metrics	Instrument for metrics only (default). If neither -metrics or -forensics is specified, it will be instrumented for metrics only.
-forensics	Instrument for forensics only.

Figure 5A

Other options	
-full	Instrument for all features; i.e., both metrics and forensics
-q	Suppress most instrumentation progress output.
-qi	Suppress already-instrumented output.
-s	Strip line number tables and local variable tables from output.
-no-manifest-update	Do not alter archive manifests of instrumented archives.
-d <out-dir>	Place instrumented classes/jars and .tj1 files into a directory tree starting at <out-dir>. Defaults to the current working directory.
-label <label>	Associates class file with a given string label.
-log <log-file>	Log instrumentation progress in <log-file>.
-p <policy.opt>	Instrument classes and jar files according to the policies found in the <policy.opt> file.
-j <map-file.jar>	Places mapfiles in <map-file.jar>.
-sourcepath <source-path>	Stores given path to Java source files in mapfiles.
-batch <batch-file>	Specifies a file containing a batch of files to instrument, one filename per line.
-bs <backup-suffix>	Specifies a suffix for backing up original class files (default is ".bak").
-ts <MMddyyyyHH:mm:ss>	Instrument only those classes modified after the specified time.
-tf <MM dd DD yy yyyy HH hh mm ss>	Pass an alternate time format to use with '-ts', suitable for the Java SimpleDateFormat class.
-r <dir>	Instrument all class and jar files in and below <dir>.

Figure 5B

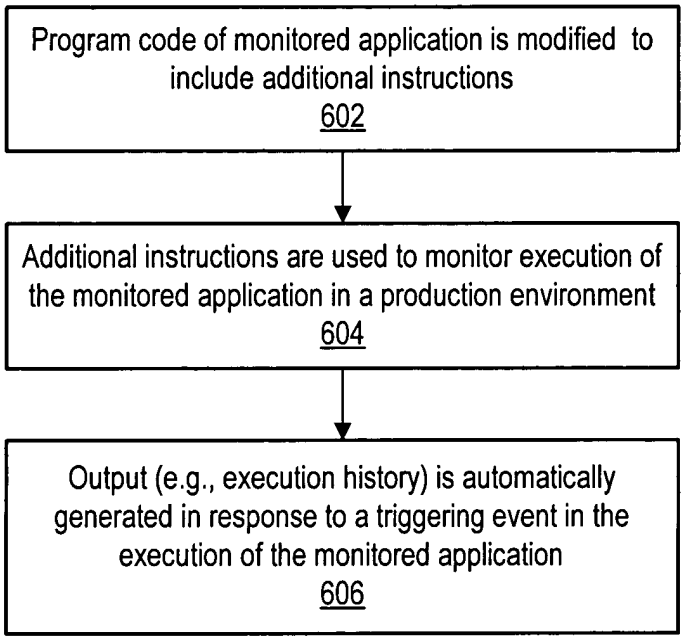

Figure 6

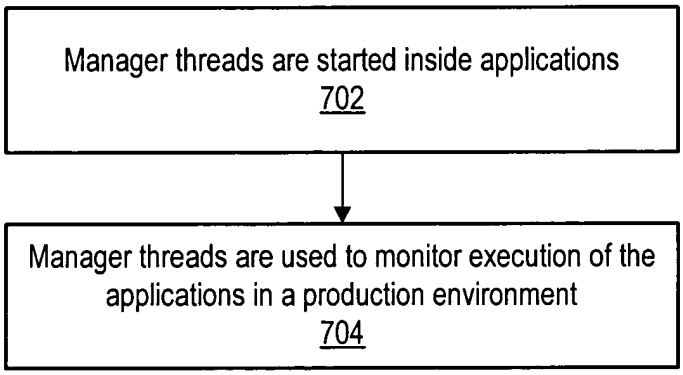

Figure 7

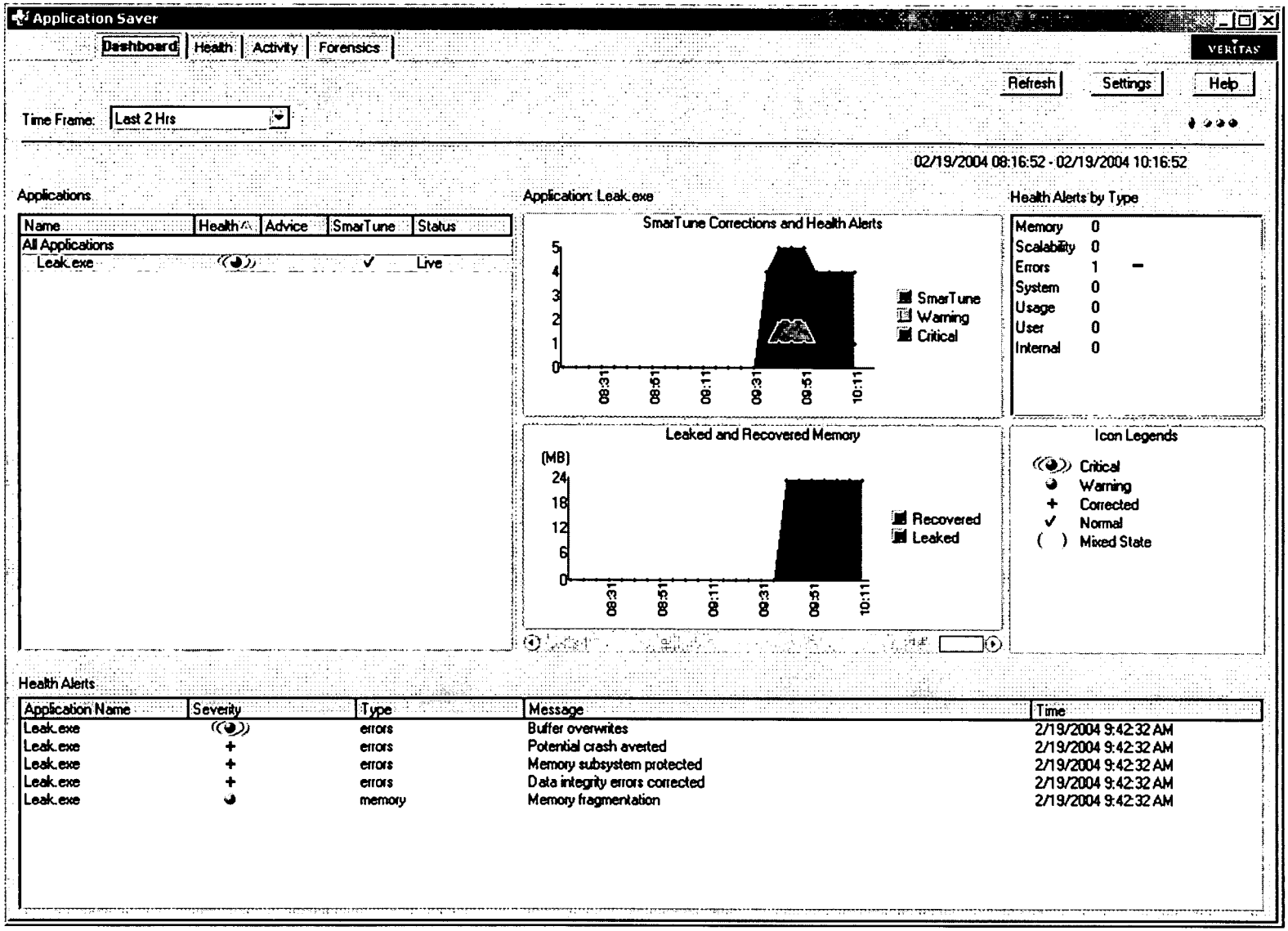
Application Saver
Dashboard
Health
Activity
Forensics
VERITAS
Refresh
Settings
Help
Time Frame:
Last 2 Hrs
02/19/2004 08:16:52 - 02/19/2004 10:16:52
Applications
Name
Health
Advice
SmarTune
Status
All Applications
Leak.exe
Live
Application: Leak.exe
SmarTune Corrections and Health Alerts
SmarTune
Warning
Critical
Leaked and Recovered Memory
(MB)
Recovered
Leaked
Health Alerts by Type
Memory 0
Scalability 0
Errors 1
System 0
Usage 0
User 0
Internal 0
Icon Legends
Critical
Warning
Corrected
Normal
Mixed State
Health Alerts
Application Name
Severity
Type
Message
Time
Leak.exe errors Buffer overwrites 2/19/2004 9:42:32 AM
Leak.exe errors Potential crash averted 2/19/2004 9:42:32 AM
Leak.exe errors Memory subsystem protected 2/19/2004 9:42:32 AM
Leak.exe errors Data integrity errors corrected 2/19/2004 9:42:32 AM
Leak.exe memory Memory fragmentation 2/19/2004 9:42:32 AM

Figure 8

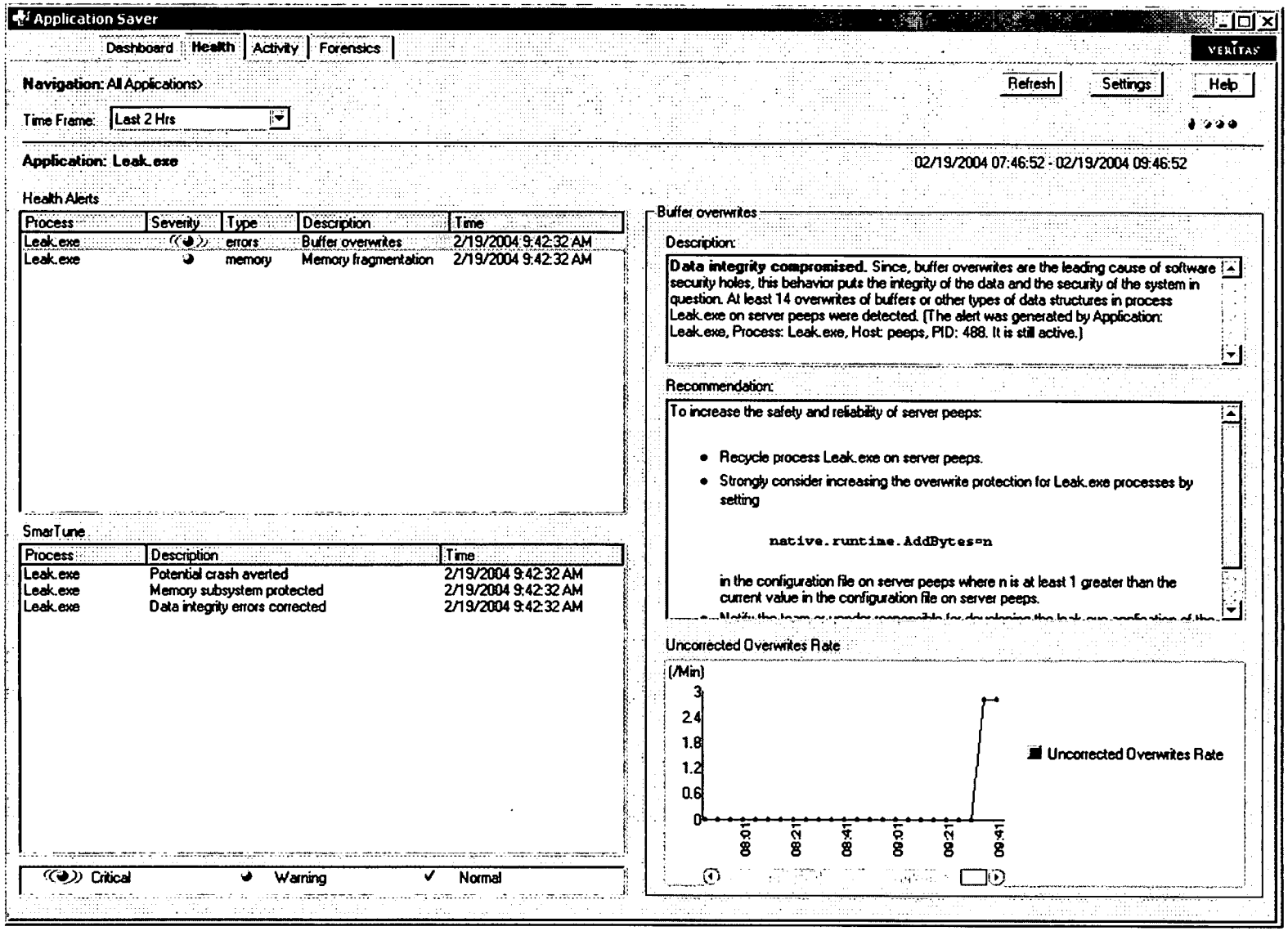
Application Saver
Dashboard
Health
Activity
Forensics
VERITAS
Navigation: All Applications>
Refresh
Settings
Help
Time Frame:
Last 2 Hrs
Application: Leak.exe
02/19/2004 07:46:52 - 02/19/2004 09:46:52
Health Alerts
Process
Severity
Type
Description
Time
Leak.exe
errors
Buffer overwrites
2/19/2004 9:42:32 AM
Leak.exe
memory
Memory fragmentation
2/19/2004 9:42:32 AM
SmarTune
Process
Description
Time
Leak.exe
Potential crash averted
2/19/2004 9:42:32 AM
Leak.exe
Memory subsystem protected
2/19/2004 9:42:32 AM
Leak.exe
Data integrity errors corrected
2/19/2004 9:42:32 AM
Critical
Warning
Normal
Buffer overwrites
Description:
Data integrity compromised. Since, buffer overwrites are the leading cause of software security holes, this behavior puts the integrity of the data and the security of the system in question. At least 14 overwrites of buffers or other types of data structures in process Leak.exe on server peeps were detected. (The alert was generated by Application: Leak.exe, Process: Leak.exe, Host: peeps, PID: 488. It is still active.)
Recommendation:
To increase the safety and reliability of server peeps:
Recycle process Leak.exe on server peeps.
Strongly consider increasing the overwrite protection for Leak.exe processes by setting
native.runtime.AddBytes=n
in the configuration file on server peeps where n is at least 1 greater than the current value in the configuration file on server peeps.
Uncorrected Overwrites Rate
(/Min)
3
2.4
1.8
1.2
0.6
0
08:01
08:21
08:41
09:01
09:21
09:41
Uncorrected Overwrites Rate

Figure 9

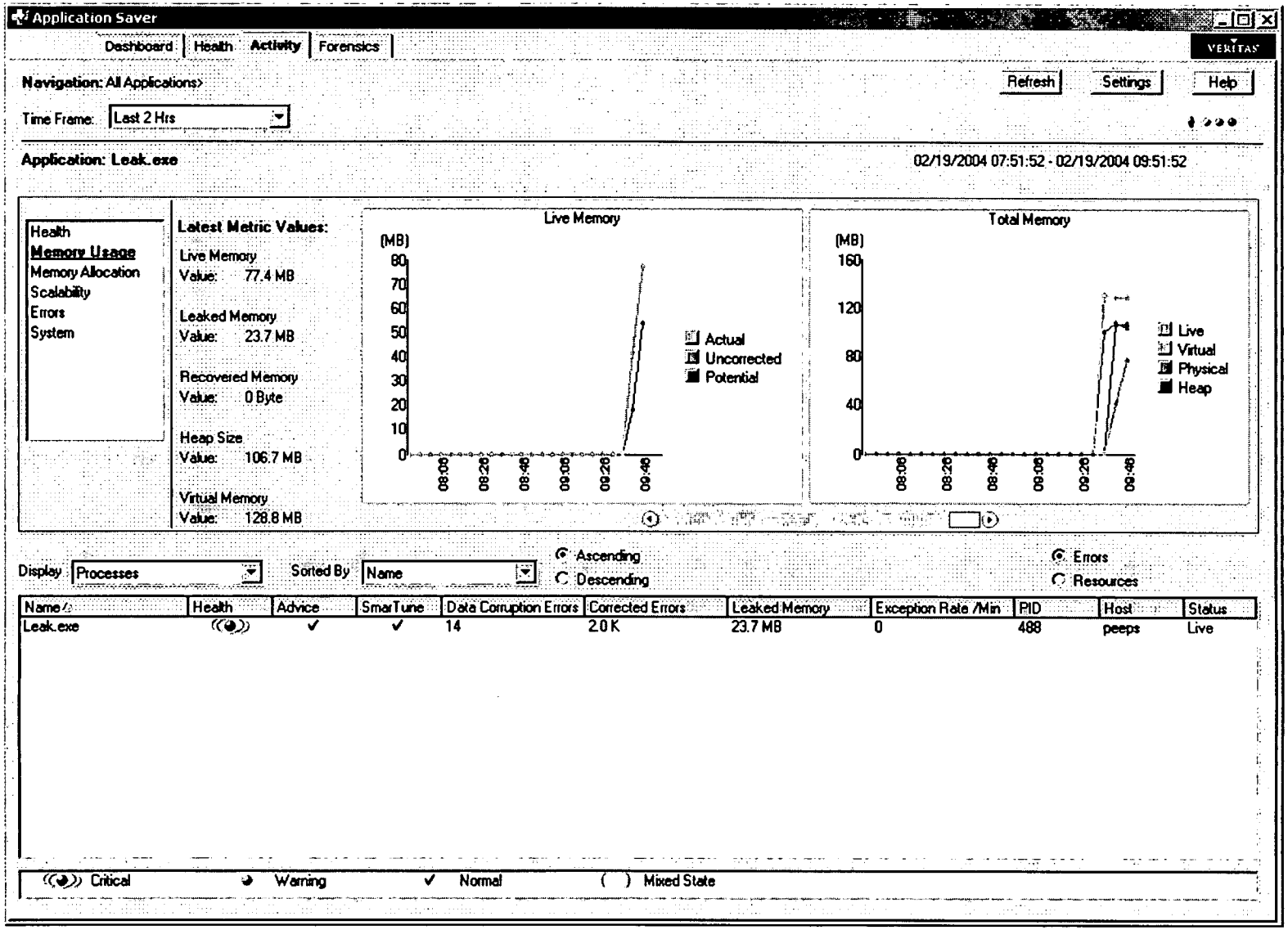
Application Saver
Dashboard
Health
Activity
Forensics
VERITAS
Navigation: All Applications>
Refresh
Settings
Help
Time Frame:
Last 2 Hrs
Application: Leak.exe
02/19/2004 07:51:52 - 02/19/2004 09:51:52
Health
Memory Usage
Memory Allocation
Scalability
Errors
System
Latest Metric Values:
Live Memory
Value: 77.4 MB
Leaked Memory
Value: 23.7 MB
Recovered Memory
Value: 0 Byte
Heap Size
Value: 106.7 MB
Virtual Memory
Value: 128.8 MB
Live Memory
(MB)
Actual
Uncorrected
Potential
Total Memory
(MB)
Live
Virtual
Physical
Heap
Display
Processes
Sorted By
Name
Ascending
Descending
Errors
Resources
Name
Health
Advice
SmarTune
Data Corruption Errors
Corrected Errors
Leaked Memory
Exception Rate /Min
PID
Host
Status
Leak.exe
14
2.0 K
23.7 MB
0
488
peeps
Live
Critical
Warning
Normal
Mixed State

Figure 10

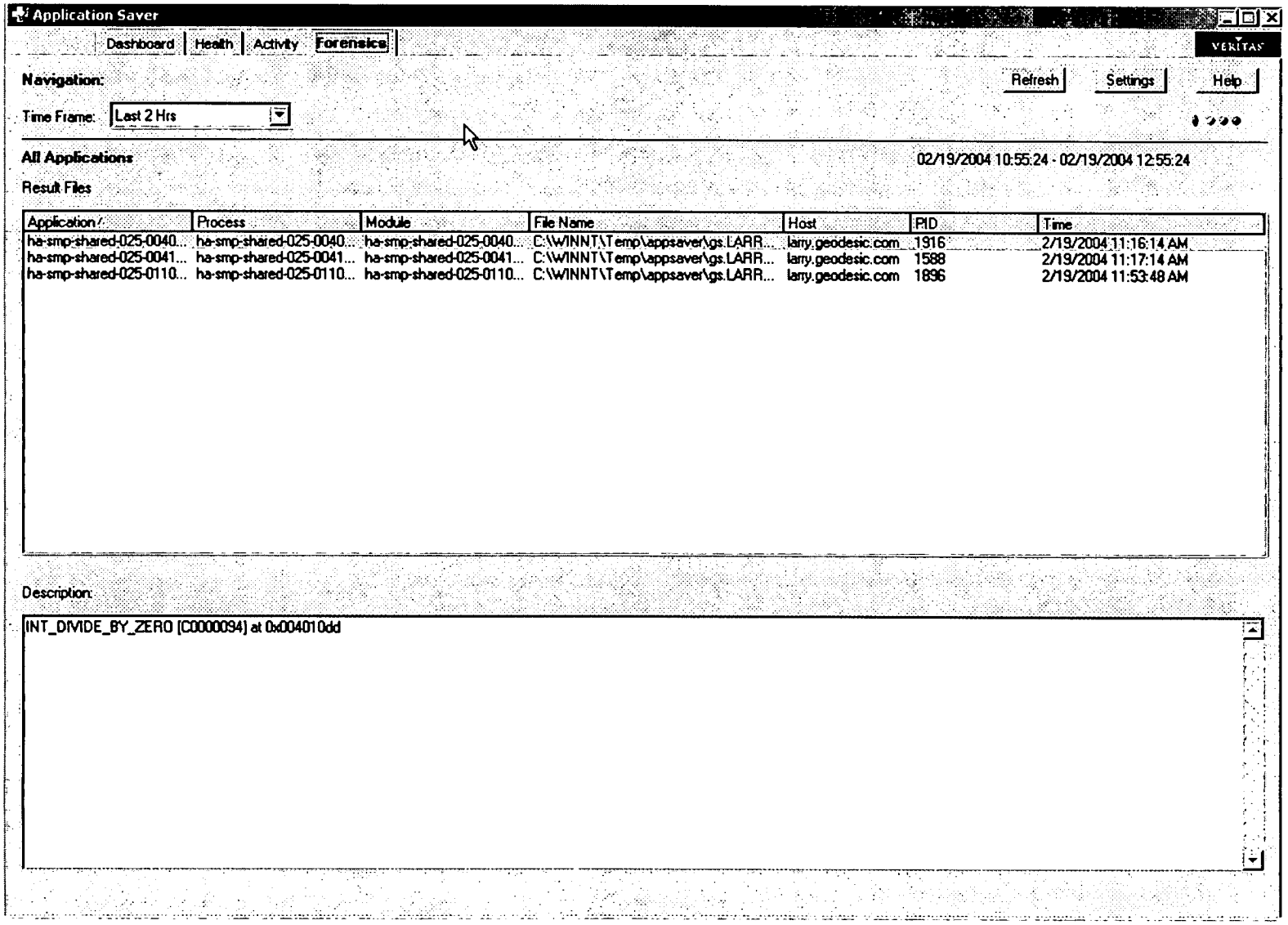
Application Saver
Dashboard
Health
Activity
Forensics
VERITAS
Navigation:
Refresh
Settings
Help
Time Frame:
Last 2 Hrs
All Applications
02/19/2004 10:55:24 - 02/19/2004 12:55:24
Result Files
Application /
Process
Module
File Name
Host
PID
Time
ha-smp-shared-025-0040...
ha-smp-shared-025-0040...
ha-smp-shared-025-0040...
C:\WINNT\Temp\appsaver\gs.LARR...
larry.geodesic.com
1916
2/19/2004 11:16:14 AM
ha-smp-shared-025-0041...
ha-smp-shared-025-0041...
ha-smp-shared-025-0041...
C:\WINNT\Temp\appsaver\gs.LARR...
larry.geodesic.com
1588
2/19/2004 11:17:14 AM
ha-smp-shared-025-0110...
ha-smp-shared-025-0110...
ha-smp-shared-025-0110...
C:\WINNT\Temp\appsaver\gs.LARR...
larry.geodesic.com
1896
2/19/2004 11:53:48 AM
Description:
INT_DIVIDE_BY_ZERO [C0000094] at 0x004010dd

Figure 11